Lunar Capstone Research Report

Akash Desai, Alan Wu, Austin Sun, Gary Yong, James Lee

Table of Contents

Abstract

This study investigated the practicality of using multivariate statistical analysis for the development of a portfolio for crypto assets. The algorithm used in this experiment is an iterative process of creating 1,000 unique portfolios and returning the optimal set of solutions. The model considers the varying risk preferences of investors and returns the leading portfolio from each of the five risk levels. Because of the randomness occurring at each iteration, we conducted a series of tests to obtain the range of expected return and drawdown of portfolios at each risk level. After 20 trials, we observed that the drawdown over the past 12 months ranged between -59.40% and -49.28%, with the average drawdown equal to -53.50%. The historical 1-year return showed larger deviations, ranging between +66.76% and +197.08%, with average return of +116.97%. Peak return ranged between +155.93% and +386.93%, with average return of +250.78%. Observations show that there is more variability among 1-year and peak returns, while drawdown is generally consistent for all risk levels. The results of this study suggest that increasing volatility tends to amplify the magnitude of a portfolio's positive return but does not have a significant impact during periods of decline.

Introduction

This research report outlines the design approaches and methods used for implementing the cryptocurrency portfolio builder tool. There will be a brief explanation on the foundation of our research on risk analysis and portfolio diversification. Following the background, the algorithm design will go through a summarized flow of the entire process, starting with data gathering and ending on how the output of the algorithm will be presented to the user. The implementation section of the report contains a step-by-step walkthrough of the program and the techniques used for statistical analysis and data transformation. The report will be concluded by a visualization of the comparison between various portfolios and a discussion on the feasibility and efficiency of the presented algorithm.

Background

The fundamental basis of the solution utilizes modern portfolio theory (MPT), a mathematical algorithm that seeks to maximize the return of an investment portfolio with minimal risk. MPT was initially introduced in 1952 by American economist Harry Markowitz and has been widely used ever since due to its flexibility with balancing various portfolio asset types and interpretability of risk analysis.

MPT, also known as mean-variance analysis, analyzes the expected return and standard deviation of portfolios using historical data of the assets included. It allows investment decisions to be made by considering the risk and return of each portfolio and choosing on the optimal solution.

From a pool of over 10,000 assets, we use a ranking model that considers not only the historical returns of assets, but also the technology stack, general sentiment, market adoption, asset performance, and liquidity to rank and pick the candidates. The following experiment uses multivariate mean-variance analysis to analyze the historical daily returns and the interrelationship between 17 crypto assets that are currently the most highly rated on the ranking system. Namely, this study will analyze and generate a diversified portfolio of Aave (AAVE), Cardano(ADA), Axie Infinity (AXS), Bitcoin (BTC), Curve DAO Token

(CRV), Polkadot (DOT), Ethereum (ETH), FTX Token (FTT), Kyber Network Crystal v2 (KNC), Chainlink (LINK), Litecoin (LTC), Polygon (MATIC), The Sandbox (SAND), Uniswap (UNI), Stellar (XLM), XRP (XRP), and Tezos (XTZ).

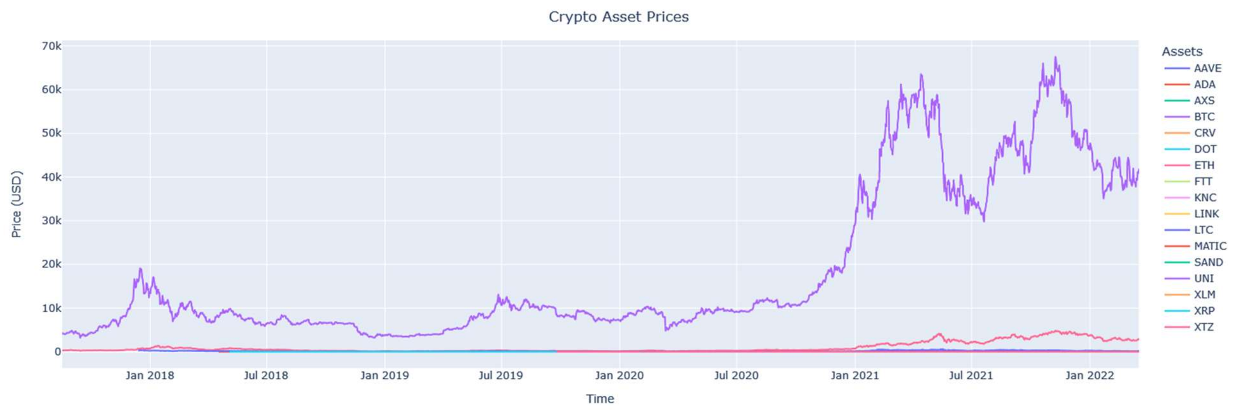


Figure 1. Historical Prices of Crypto Assets

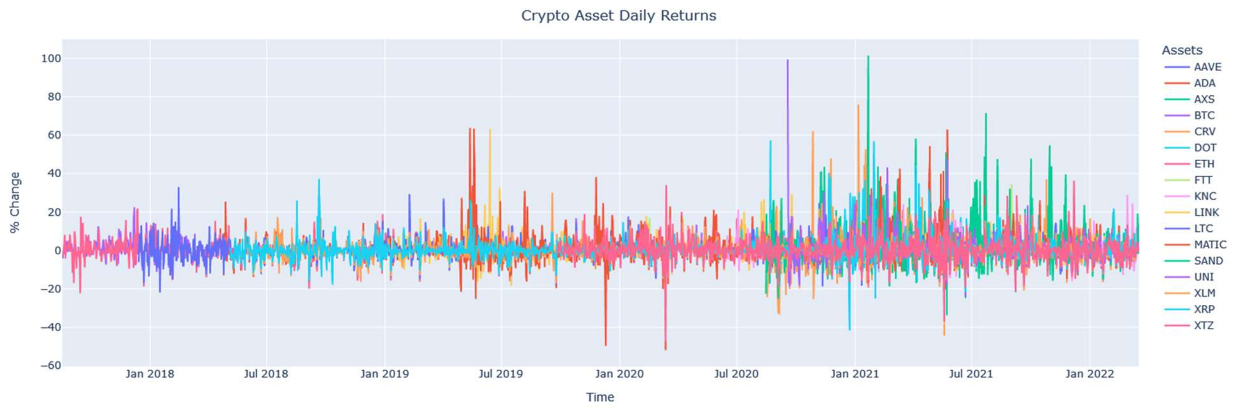


Figure 2. Historical Daily Returns of Crypto Assets

Algorithm Design

The algorithm is designed with a focus on efficiency and simplicity. However, it is effective in determining the expected return and volatility of each portfolio. The initial step is to obtain the daily closing prices of each asset and aggregating the values to get monthly returns. We have set a monthly geometric decay of 5% to put more weight on recent data. Figure 2 is an overview of the past several years of daily returns for each of the 17 assets. The primary reason for putting less weight on older data is in consideration of the declining volatility and stabilizing conditions of the crypto market. However, it is important to note that fundamentals may change at any point and require a modification of the current model.

The weighted average return for each asset is obtained and used to derive additional measures such as standard deviation, correlation, and covariance. These measures are used as intermediate steps of the process of calculating expected portfolio return and standard deviation, which are the two attributes of a given portfolio that allow us to determine optimality.

Obtaining the optimal portfolio requires an iterative process through random number generation. Through a series of tests, we have decided to set the number of generated portfolios to 1000, in consideration of runtime. Our algorithm uses random numbers to produce allocation weights of assets in each iteration. In other words, a unique portfolio is created in every iteration. The unique weights of each portfolio's assets are then used to determine the associated risk and expected return of the portfolio.

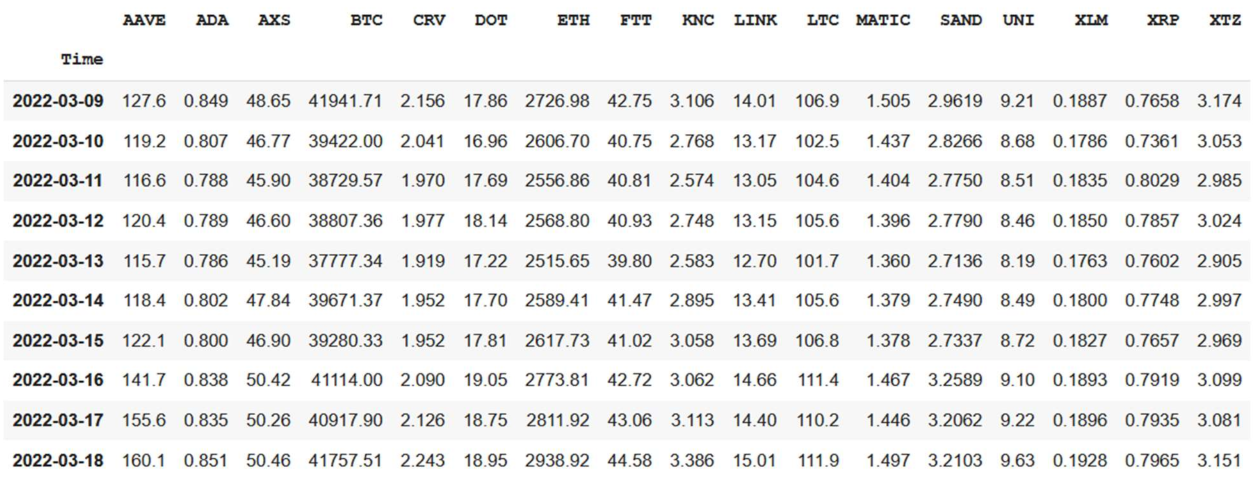
One important point to consider is a specific user's risk tolerance. A risk-averse investor will likely favour a portfolio with minimal risk even though its expected return is comparably low. On the other hand, a risk-tolerant investor may prefer an aggressive portfolio with high risk and high return. Taking this into account, we have decided to implement risk buckets that are separated by volatility ranges. Following the iterations, the randomly generated portfolios are assigned to one of the five risk buckets based on portfolio standard deviation.

The last step of the algorithm is the decision-making process. For each risk bucket, we obtain the Sharpe ratios of each portfolio within. The Sharpe ratio is a type of measurement using the excess return of an investment and the underlying volatility, indicating the relative performance of the given portfolio. The portfolios are then ranked based on Sharpe ratio in descending order to rearrange the choices in each bucket. Finally, the top portfolio in each risk bucket is selected as the optimal solution for the given risk level. On the application interface, the user will be suggested with one of the five portfolios depending on his or her risk preference.

Implementation

A custom function is used to fetch the daily closing prices of all the assets. The resulting data frames is concatenated into one single data frame to be used for the remainder of the algorithm.

```
years = 5
start = datetime.today() - relativedelta(years=years)
dfs = []
for x in assets:
  try:
    dfs.append(ohlcv(start, x, '1d'))
  except:
    pass
data = pd.concat(dfs, axis=1)
data = data.reindex(sorted(data.columns), axis=1)
data = data[:-1]
```

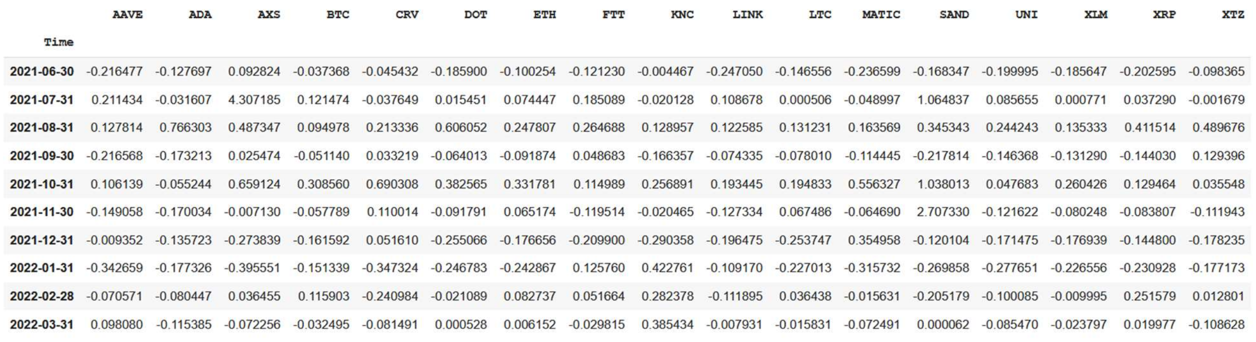


Time	AAVE	ADA	AXS	BTC	CRV	DOT	ETH	FTT	KNC	LINK	LTC	MATIC	SAND	UNI	XLM	XRP	XTZ
2022-03-09	127.6	0.849	48.65	41941.71	2.156	17.86	2726.98	42.75	3.106	14.01	106.9	1.505	2.9619	9.21	0.1887	0.7658	3.174
2022-03-10	119.2	0.807	46.77	39422.00	2.041	16.96	2606.70	40.75	2.768	13.17	102.5	1.437	2.8266	8.68	0.1786	0.7361	3.053
2022-03-11	116.6	0.788	45.90	38729.57	1.970	17.69	2556.86	40.81	2.574	13.05	104.6	1.404	2.7750	8.51	0.1835	0.8029	2.985
2022-03-12	120.4	0.789	46.60	38807.36	1.977	18.14	2568.80	40.93	2.748	13.15	105.6	1.396	2.7790	8.46	0.1850	0.7857	3.024
2022-03-13	115.7	0.786	45.19	37777.34	1.919	17.22	2515.65	39.80	2.583	12.70	101.7	1.360	2.7136	8.19	0.1763	0.7602	2.905
2022-03-14	118.4	0.802	47.84	39671.37	1.952	17.70	2589.41	41.47	2.895	13.41	105.6	1.379	2.7490	8.49	0.1800	0.7748	2.997
2022-03-15	122.1	0.800	46.90	39280.33	1.952	17.81	2617.73	41.02	3.058	13.69	106.8	1.378	2.7337	8.72	0.1827	0.7657	2.969
2022-03-16	141.7	0.838	50.42	41114.00	2.090	19.05	2773.81	42.72	3.062	14.66	111.4	1.467	3.2589	9.10	0.1893	0.7919	3.099
2022-03-17	155.6	0.835	50.26	40917.90	2.126	18.75	2811.92	43.06	3.113	14.40	110.2	1.446	3.2062	9.22	0.1896	0.7935	3.081
2022-03-18	160.1	0.851	50.46	41757.51	2.243	18.95	2938.92	44.58	3.386	15.01	111.9	1.497	3.2103	9.63	0.1928	0.7965	3.151

Figure 3. Recent Closing Prices of Assets

With the closing prices saved to the data frame, the daily and monthly returns are calculated with the respective decay rates.

```
# Monthly decay rate
m_decay_rate = 0.05

# Get number of months since start_date
def get_months(start_date):
  end_date = date.today()
  years = end_date.year - start_date.year
  months = end_date.month - start_date.month
  return years * 12 + months

monthly_closes = data.resample('M').last() # Resample daily returns to monthly
monthly_returns = monthly_closes.pct_change() # Monthly returns
monthly_returns = monthly_returns.apply(lambda x: x * (1-
m_decay_rate) ** get_months(x.name), axis=1) # Monthly returns with decay
```

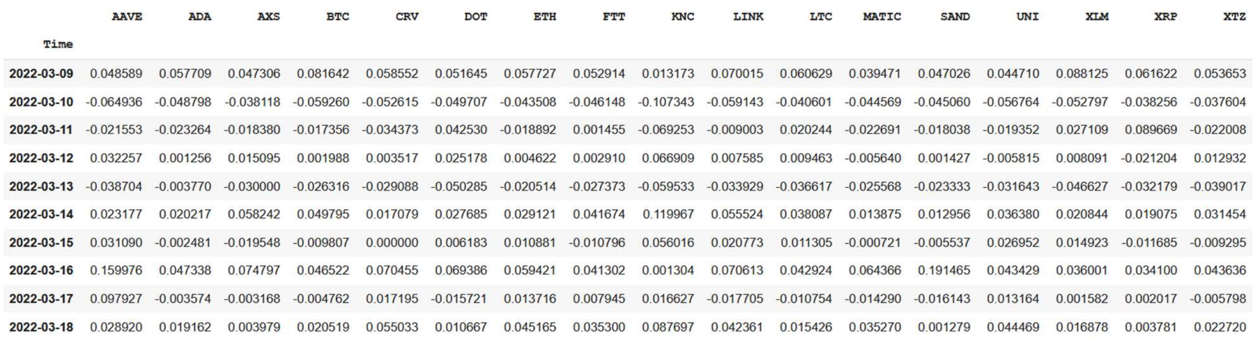


Time	AAVE	ADA	AXS	BTC	CRV	DOT	ETH	FTT	KNC	LINK	LTC	MATIC	SAND	UNI	XLM	XRP	XTZ
2021-06-30	-0.216477	-0.127697	0.092824	-0.037368	-0.045432	-0.185900	-0.100254	-0.121230	-0.004467	-0.247050	-0.146556	-0.236599	-0.168347	-0.199995	-0.185647	-0.202595	-0.098365
2021-07-31	0.211434	-0.031607	4.307185	0.121474	-0.037649	0.015451	0.074447	0.185089	-0.020128	0.108678	0.000506	-0.048997	1.064837	0.085655	0.000771	0.037290	-0.001679
2021-08-31	0.127814	0.766303	0.487347	0.094978	0.213336	0.606052	0.247807	0.264688	0.128957	0.122585	0.131231	0.163569	0.345343	0.244243	0.135333	0.411514	0.489676
2021-09-30	-0.216568	-0.173213	0.025474	-0.051140	0.033219	-0.064013	-0.091874	0.048683	-0.166357	-0.074335	-0.078010	-0.114445	-0.217814	-0.146368	-0.131290	-0.144030	0.129396
2021-10-31	0.106139	-0.055244	0.659124	0.308560	0.690308	0.382565	0.331781	0.114989	0.256891	0.193445	0.194833	0.556327	1.038013	0.047683	0.260426	0.129464	0.035548
2021-11-30	-0.149058	-0.170034	-0.007130	-0.057789	0.110014	-0.091791	0.065174	-0.119514	-0.020465	-0.127334	0.067486	-0.064690	2.707330	-0.121622	-0.080248	-0.083807	-0.111943
2021-12-31	-0.009352	-0.135723	-0.273839	-0.161592	0.051610	-0.255066	-0.176656	-0.209900	-0.290358	-0.196475	-0.253747	0.354958	-0.120104	-0.171475	-0.176939	-0.144800	-0.178235
2022-01-31	-0.342659	-0.177326	-0.395551	-0.151339	-0.347324	-0.246783	-0.242867	0.125760	0.422761	-0.109170	-0.227013	-0.315732	-0.269858	-0.277651	-0.226556	-0.230928	-0.177173
2022-02-28	-0.070571	-0.080447	0.036455	0.115903	-0.240984	-0.021089	0.082737	0.051664	0.282378	-0.111895	0.036438	-0.015631	-0.205179	-0.100085	-0.009995	0.251579	0.012801
2022-03-31	0.098080	-0.115385	-0.072256	-0.032495	-0.081491	0.000528	0.006152	-0.029815	0.385434	-0.007931	-0.015831	-0.072491	0.000062	-0.085470	-0.023797	0.019977	-0.108628

Figure 4. Recent Monthly Returns

There are two methods of determining the monthly standard deviation of a portfolio. The first method uses daily returns to initially obtain the daily standard deviation and then scale and transform the result into monthly standard deviation. The second method uses monthly returns to directly calculate the monthly standard deviation. To improve the accuracy and precision of standard deviation estimations, the first method is chosen.

```
# Daily decay rate
d_decay_rate = 1 - (1-m_decay_rate)**(1/30)

# Get number of days since start_date
def get_days(start_date):
  end_date = date.today()
  diff = end_date - start_date.date()
  return diff.days

daily_returns = data.pct_change() # Daily returns
daily_returns = daily_returns.apply(lambda x: x * (1-d_de-
cay_rate) ** get_days(x.name), axis=1) # Daily returns with decay
```



	AAVE	ADA	AXS	BTC	CRV	DOT	ETH	FTT	KNC	LINK	LTC	MATIC	SAND	UNI	XLM	XRP	XTZ
Time																	
2022-03-09	0.048589	0.057709	0.047306	0.081642	0.058552	0.051645	0.057727	0.052914	0.013173	0.070015	0.060629	0.039471	0.047026	0.044710	0.088125	0.061622	0.053653
2022-03-10	-0.064936	-0.048798	-0.038118	-0.059260	-0.052615	-0.049707	-0.043508	-0.046148	-0.107343	-0.059143	-0.040601	-0.044569	-0.045060	-0.056764	-0.052797	-0.038256	-0.037604
2022-03-11	-0.021553	-0.023264	-0.018380	-0.017356	-0.034373	0.042530	-0.018892	0.001455	-0.069253	-0.009003	0.020244	-0.022691	-0.018038	-0.019352	0.027109	0.089669	-0.022008
2022-03-12	0.032257	0.001256	0.015095	0.001988	0.003517	0.025178	0.004622	0.002910	0.066909	0.007585	0.009463	-0.005640	0.001427	-0.005815	0.008091	-0.021204	0.012932
2022-03-13	-0.038704	-0.003770	-0.030000	-0.026316	-0.029088	-0.050285	-0.020514	-0.027373	-0.059533	-0.033929	-0.036617	-0.025568	-0.023333	-0.031643	-0.046627	-0.032179	-0.039017
2022-03-14	0.023177	0.020217	0.058242	0.049795	0.017079	0.027685	0.029121	0.041674	0.119967	0.055524	0.038087	0.013875	0.012956	0.036380	0.020844	0.019075	0.031454
2022-03-15	0.031090	-0.002481	-0.019548	-0.009807	0.000000	0.006183	0.010881	-0.010796	0.056016	0.020773	0.011305	-0.000721	-0.005537	0.026952	0.014923	-0.011685	-0.009295
2022-03-16	0.159976	0.047338	0.074797	0.046522	0.070455	0.069386	0.059421	0.041302	0.001304	0.070613	0.042924	0.064366	0.191465	0.043429	0.036001	0.034100	0.043636
2022-03-17	0.097927	-0.003574	-0.003168	-0.004762	0.017195	-0.015721	0.013716	0.007945	0.016627	-0.017705	-0.010754	-0.014290	-0.016143	0.013164	0.001582	0.002017	-0.005798
2022-03-18	0.028920	0.019162	0.003979	0.020519	0.055033	0.010667	0.045165	0.035300	0.087697	0.042361	0.015426	0.035270	0.001279	0.044469	0.016878	0.003781	0.022720

Figure 5. Recent Daily Returns

The following code snippet uses the information on daily and monthly returns to generate the weighted average monthly return and monthly standard deviation of all 17 assets. Using these two values and a risk-free rate of 0.1%, the Sharpe ratio can be determined.

Monthly Standard Deviation (30-day approximation):

$$\sigma = \sqrt{\frac{\sum(x_i - \mu)^2}{N}} \cdot \sqrt{30}$$

x_i = Individual daily return

μ = Average daily return

N = Number of data points

Sharpe Ratio:

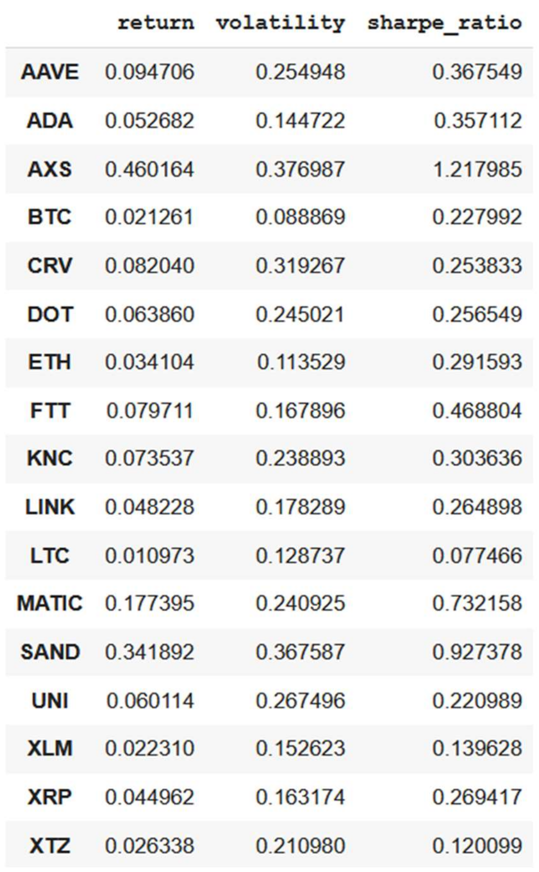
$$S = \frac{E[R_p - R_f]}{\sigma_p}$$

R_p = Portfolio return

R_f = Risk-free rate

σ_p = Standard deviation of portfolio

```
avg_return = monthly_returns.mean() # Average monthly return
std_dev = daily_returns.std()*np.sqrt(30) # Monthly standard deviation

risk_free_rate = 0.001 # risk-free rate (0.1%)

ret_vol_sr = pd.concat([avg_return, std_dev], axis=1)
ret_vol_sr.columns = ['return','volatility']
ret_vol_sr['sharpe_ratio'] = ret_vol_sr.apply(lambda x: (x['return'] - risk_free_rate)
/x['volatility'], axis=1) # Calculate Sharpe ratio
ret_vol_sr = ret_vol_sr.applymap(lambda x: round(x,6))
```



	return	volatility	sharpe_ratio
AAVE	0.094706	0.254948	0.367549
ADA	0.052682	0.144722	0.357112
AXS	0.460164	0.376987	1.217985
BTC	0.021261	0.088869	0.227992
CRV	0.082040	0.319267	0.253833
DOT	0.063860	0.245021	0.256549
ETH	0.034104	0.113529	0.291593
FTT	0.079711	0.167896	0.468804
KNC	0.073537	0.238893	0.303636
LINK	0.048228	0.178289	0.264898
LTC	0.010973	0.128737	0.077466
MATIC	0.177395	0.240925	0.732158
SAND	0.341892	0.367587	0.927378
UNI	0.060114	0.267496	0.220989
XLM	0.022310	0.152623	0.139628
XRP	0.044962	0.163174	0.269417
XTZ	0.026338	0.210980	0.120099

Figure 6. Weighted Return, Volatility, and Sharpe Ratio of All Assets

To begin the iterations, there is one more piece of information that is required. The covariance matrix allows us to calculate portfolio variance. It will be used in conjunction with randomly generated weights on each iteration.

Covariance:

$$Cov(X, Y) = \frac{\sum(X_i - \bar{X})(Y_i - \bar{Y})}{N}$$

X_i = Individual return of asset X

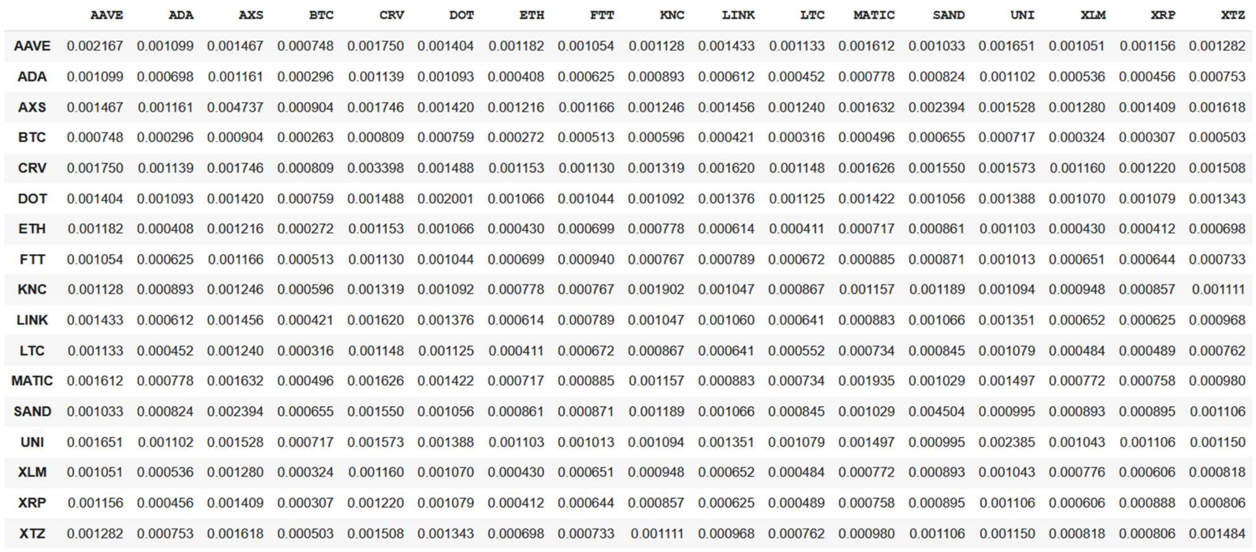
$\bar{X}$ = Average return of asset X

Y_i = Individual return of asset Y

$\bar{Y}$ = Average return of asset Y

N = Number of data points

```
cov_matrix = daily_returns.cov()
```



	AAVE	ADA	AXS	BTC	CRV	DOT	ETH	FTT	KNC	LINK	LTC	MATIC	SAND	UNI	XLM	XRP	XTZ
AAVE	0.002167	0.001099	0.001467	0.000748	0.001750	0.001404	0.001182	0.001054	0.001128	0.001433	0.001133	0.001612	0.001033	0.001651	0.001051	0.001156	0.001282
ADA	0.001099	0.000698	0.001161	0.000296	0.001139	0.001093	0.000408	0.000625	0.000893	0.000612	0.000452	0.000778	0.000824	0.001102	0.000536	0.000456	0.000753
AXS	0.001467	0.001161	0.004737	0.000904	0.001746	0.001420	0.001216	0.001166	0.001246	0.001456	0.001240	0.001632	0.002394	0.001528	0.001280	0.001409	0.001618
BTC	0.000748	0.000296	0.000904	0.000263	0.000809	0.000759	0.000272	0.000513	0.000596	0.000421	0.000316	0.000496	0.000655	0.000717	0.000324	0.000307	0.000503
CRV	0.001750	0.001139	0.001746	0.000809	0.003398	0.001488	0.001153	0.001130	0.001319	0.001620	0.001148	0.001626	0.001550	0.001573	0.001160	0.001220	0.001508
DOT	0.001404	0.001093	0.001420	0.000759	0.001488	0.002001	0.001066	0.001044	0.001092	0.001376	0.001125	0.001422	0.001056	0.001388	0.001070	0.001079	0.001343
ETH	0.001182	0.000408	0.001216	0.000272	0.001153	0.001066	0.000430	0.000699	0.000778	0.000614	0.000411	0.000717	0.000861	0.001103	0.000430	0.000412	0.000698
FTT	0.001054	0.000625	0.001166	0.000513	0.001130	0.001044	0.000699	0.000940	0.000767	0.000789	0.000672	0.000885	0.000871	0.001013	0.000651	0.000644	0.000733
KNC	0.001128	0.000893	0.001246	0.000596	0.001319	0.001092	0.000778	0.000767	0.001902	0.001047	0.000867	0.001157	0.001189	0.001094	0.000948	0.000857	0.001111
LINK	0.001433	0.000612	0.001456	0.000421	0.001620	0.001376	0.000614	0.000789	0.001047	0.001060	0.000641	0.000883	0.001066	0.001351	0.000652	0.000625	0.000968
LTC	0.001133	0.000452	0.001240	0.000316	0.001148	0.001125	0.000411	0.000672	0.000867	0.000641	0.000552	0.000734	0.000845	0.001079	0.000484	0.000489	0.000762
MATIC	0.001612	0.000778	0.001632	0.000496	0.001626	0.001422	0.000717	0.000885	0.001157	0.000883	0.000734	0.001935	0.001029	0.001497	0.000772	0.000758	0.000980
SAND	0.001033	0.000824	0.002394	0.000655	0.001550	0.001056	0.000861	0.000871	0.001189	0.001066	0.000845	0.001029	0.004504	0.000995	0.000893	0.000895	0.001106
UNI	0.001651	0.001102	0.001528	0.000717	0.001573	0.001388	0.001103	0.001013	0.001094	0.001351	0.001079	0.001497	0.000995	0.002385	0.001043	0.001106	0.001150
XLM	0.001051	0.000536	0.001280	0.000324	0.001160	0.001070	0.000430	0.000651	0.000948	0.000652	0.000484	0.000772	0.000893	0.001043	0.000776	0.000606	0.000818
XRP	0.001156	0.000456	0.001409	0.000307	0.001220	0.001079	0.000412	0.000644	0.000857	0.000625	0.000489	0.000758	0.000895	0.001106	0.000606	0.000888	0.000806
XTZ	0.001282	0.000753	0.001618	0.000503	0.001508	0.001343	0.000698	0.000733	0.001111	0.000968	0.000762	0.000980	0.001106	0.001150	0.000818	0.000806	0.001484

Figure 7. Covariance Matrix of Assets

Portfolio variance:

$$\sigma_p^2 = \sum_{i=1}^{n} \sum_{j=1}^{n} w_i w_j Cov(a_i, a_j)$$

w_i = Weight of Asset i

a_i = Asset i

Before running the iterations, we first initialize empty lists to store the associated measures for each portfolio created in the loop.

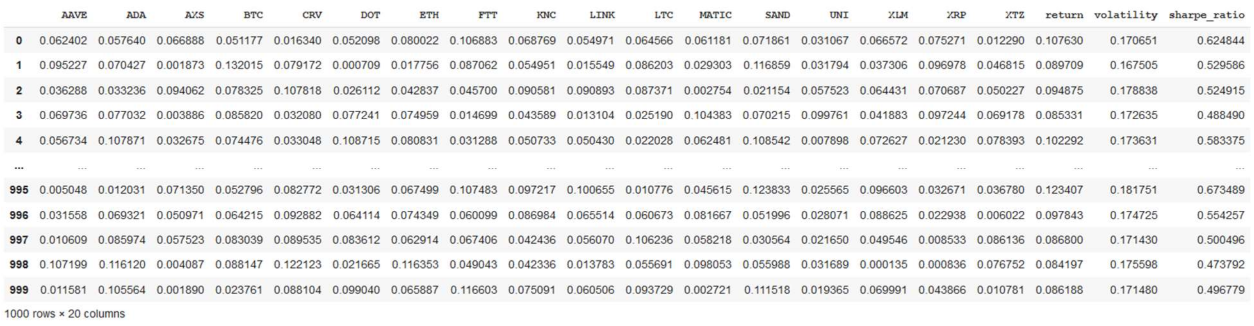
```
port_weights = [] # Asset weights
port_returns = [] # Portfolio return
port_std_dev = [] # Portfolio standard deviation
port_sharpe_ratio = [] # Portfolio Sharpe ratio

num_ports = 1000 # Number of iterations
```

Data preparation is now complete and the algorithm now proceeds to the iteration stage.

```
for port in range(num_ports):
  weights = np.random.random(len(assets)) # Generate 10 random numbers between 0 and 1
  weights = weights/np.sum(weights) # Divide each random number by sum to make 100% co
mposition
  port_weights.append(weights)

  curr_return = np.dot(weights, avg_return) # Portfolio average return
  port_returns.append(curr_return)

  var = cov_matrix.mul(weights, axis=0).mul(weights, axis=1).sum().sum() # Portfolio v
ariance
  std_dev = np.sqrt(var) * np.sqrt(30) # Portfolio standard deviation
  port_std_dev.append(std_dev)

  sharpe_ratio = (curr_return - risk_free_rate)/std_dev # Portfolio Sharpe ratio
  port_sharpe_ratio.append(sharpe_ratio)
```

The results are concatenated into one large data frame as shown below. Each row displays the precise allocations for each asset, as well as the portfolio's expected return, volatility, and Sharpe ratio.

```
df = pd.DataFrame(port_weights, columns=assets)
df['return'] = port_returns
df['volatility'] = port_std_dev
df['sharpe_ratio'] = port_sharpe_ratio
df = df.applymap(lambda x: round(x, 6))
```



	AAVE	ADA	AXS	BTC	CRV	DOT	ETH	FTT	KNC	LINK	LTC	MATIC	SAND	UNI	XLM	XRP	XTZ	return	volatility	sharpe_ratio
0	0.062402	0.057640	0.066888	0.051177	0.016340	0.052098	0.080022	0.106883	0.068769	0.054971	0.064566	0.061181	0.071861	0.031067	0.066572	0.075271	0.012290	0.107630	0.170651	0.624844
1	0.095227	0.070427	0.001873	0.132015	0.079172	0.000709	0.017756	0.087062	0.054951	0.015549	0.086203	0.029303	0.116859	0.031794	0.037306	0.096978	0.046815	0.089709	0.167505	0.529586
2	0.036288	0.033236	0.094062	0.078325	0.107818	0.026112	0.042837	0.045700	0.090581	0.090893	0.087371	0.002754	0.021154	0.057523	0.064431	0.070687	0.050227	0.094875	0.178838	0.524915
3	0.069736	0.077032	0.003886	0.085820	0.032080	0.077241	0.074959	0.014699	0.043589	0.013104	0.025190	0.104383	0.070215	0.099761	0.041883	0.097244	0.069178	0.085331	0.172635	0.488490
4	0.056734	0.107871	0.032675	0.074476	0.033048	0.108715	0.080831	0.031288	0.050733	0.050430	0.022028	0.062481	0.108542	0.007898	0.072627	0.021230	0.078393	0.102292	0.173631	0.583375
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
995	0.005048	0.012031	0.071350	0.052796	0.082772	0.031306	0.067499	0.107483	0.097217	0.100655	0.010776	0.045615	0.123833	0.025565	0.096603	0.032671	0.036780	0.123407	0.181751	0.673489
996	0.031558	0.069321	0.050971	0.064215	0.092882	0.064114	0.074349	0.060099	0.086984	0.065514	0.060673	0.081667	0.051996	0.028071	0.088625	0.022938	0.006022	0.097843	0.174725	0.554257
997	0.010609	0.085974	0.057523	0.083039	0.089535	0.083612	0.062914	0.067406	0.042436	0.056070	0.106236	0.058218	0.030564	0.021650	0.049546	0.008533	0.086136	0.086800	0.171430	0.500496
998	0.107199	0.116120	0.004087	0.088147	0.122123	0.021665	0.116353	0.049043	0.042336	0.013783	0.055691	0.098053	0.055988	0.031689	0.000135	0.000836	0.076752	0.084197	0.175598	0.473792
999	0.011581	0.105564	0.001890	0.023761	0.088104	0.099040	0.065887	0.116603	0.075091	0.060506	0.093729	0.002721	0.111518	0.019365	0.069991	0.043866	0.010781	0.086188	0.171480	0.496779

1000 rows × 20 columns

Figure 8. Portfolio Allocations, Expected Return, Volatility, and Sharpe Ratio

The following scatterplot shows the distribution of the randomly generated portfolios across varying ranges of standard deviation and expected return on the monthly scale.

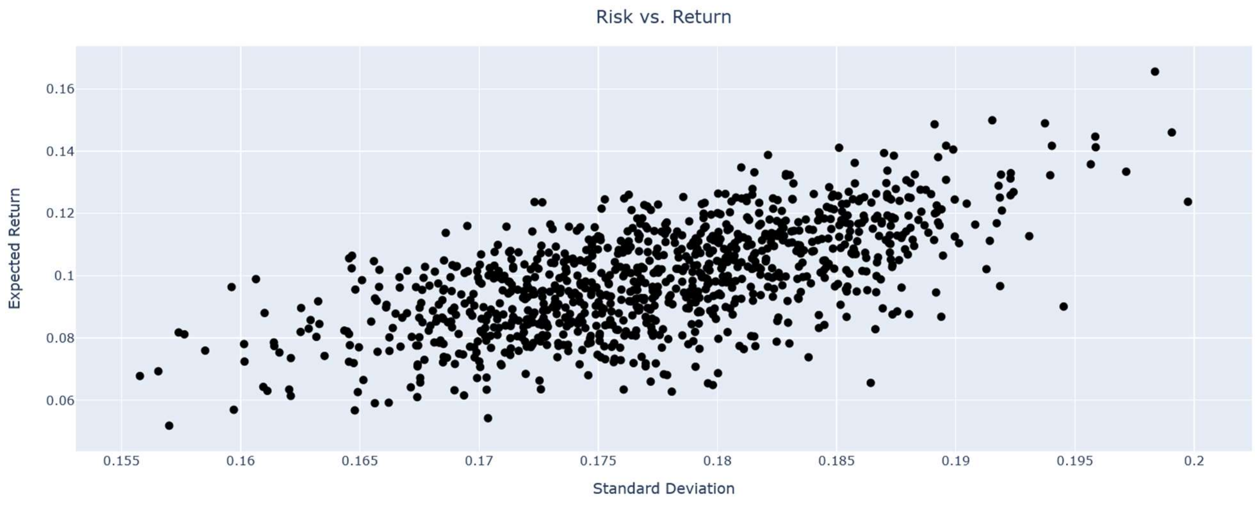


Figure 9. Risk vs. Return (Monthly Standard Deviation vs. Expected Return)

The default number of risk buckets that we have chosen is 5. Portfolios are ordered in ascending standard deviation and evenly distributed among the five risk buckets.

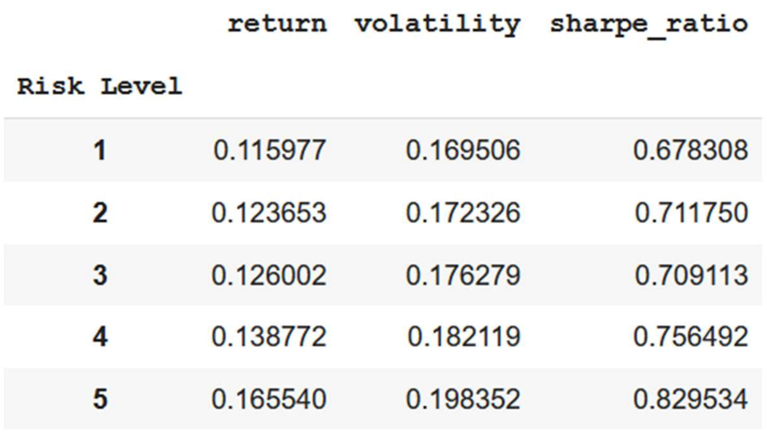
```
num_buckets = 5
df_sorted = df.sort_values(by=['volatility']) # Sort portfolios in increasing order of
 standard deviation
risk_buckets = np.array_split(df_sorted, num_buckets) # Split into 5 risk buckets
```

From each bucket, we pick out the portfolio with the highest Sharpe ratio and store other details about the assets that make up the portfolio such as allocation and expected return.

```
portfolios = [] # Store for top portfolios in each bucket

for i in range(num_buckets):
  bucket_df = risk_buckets[i]
  max_sr = bucket_df.loc[bucket_df['sharpe_ratio'].idxmax()] # Max Sharpe ratio

  assets_info = []

  # Asset Information
  for a in assets:
    assets_info.append({
      'asset': a,
      'allocation': max_sr[a],
      'return': ret_vol_sr['return'][a],
      'volatility': ret_vol_sr['volatility'][a],
      'sharpe_ratio': ret_vol_sr['sharpe_ratio'][a]
    })
  # Portfolio Information
  obj = {
    "risk_level": i+1,
    "portfolio": {
      "return": max_sr['return'],
      "volatility": max_sr['volatility'],
      "sharpe_ratio": max_sr['sharpe_ratio']
    },
    "assets": assets_info
  }

  portfolios.append(obj)
```

The details of each portfolio suggestion are summarized into a table below.

```
result = pd.DataFrame(columns=['return','volatility','sharpe_ratio'])
for i in range(num_buckets):
  bucket_port = portfolios[i]['portfolio']
  result = result.append({
      'return': bucket_port['return'],
      'volatility': bucket_port['volatility'],
      'sharpe_ratio': bucket_port['sharpe_ratio'],
  }, ignore_index=True)
result.index.name = 'Risk Level'
result.index += 1
```



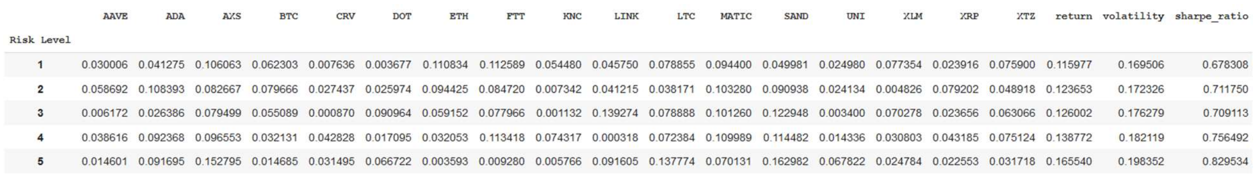
	return	volatility	sharpe_ratio
Risk Level			
1	0.115977	0.169506	0.678308
2	0.123653	0.172326	0.711750
3	0.126002	0.176279	0.709113
4	0.138772	0.182119	0.756492
5	0.165540	0.198352	0.829534

Figure 10. Summary of Leading Portfolios

Results

The results table from Figure 10 is extended to show the exact allocations of assets in each portfolio. Figure 11 shows the complete information that will be used on the application interface to provide users with details of the portfolio of interest.

```
result_cols = assets + ['return','volatility','sharpe_ratio']
result = pd.DataFrame(columns=result_cols)
for i in range(num_buckets):
  bucket = portfolios[i]
  bucket_port = bucket['portfolio']
  allocations = [x['allocation'] for x in portfolios[i]['assets']] + [bucket_port['ret
urn'], bucket_port['volatility'], bucket_port['sharpe_ratio']]
  result.loc[len(result)] = allocations
result.index.name = 'Risk Level'
result.index += 1
```



	AAVE	ADA	AXS	BTC	CRV	DOT	ETH	FTT	KNC	LINK	LTC	MATIC	SAND	UNI	XLM	XRP	XTZ	return	volatility	sharpe_ratio
Risk Level																				
1	0.030006	0.041275	0.106063	0.062303	0.007636	0.003677	0.110834	0.112589	0.054480	0.045750	0.078855	0.094400	0.049981	0.024980	0.077354	0.023916	0.075900	0.115977	0.169506	0.678308
2	0.058692	0.108393	0.082667	0.079666	0.027437	0.025974	0.094425	0.084720	0.007342	0.041215	0.038171	0.103280	0.090938	0.024134	0.004826	0.079202	0.048918	0.123653	0.172326	0.711750
3	0.006172	0.026386	0.079499	0.055089	0.000870	0.090964	0.059152	0.077966	0.001132	0.139274	0.078888	0.101260	0.122948	0.003400	0.070278	0.023656	0.063066	0.126002	0.176279	0.709113
4	0.038616	0.092368	0.096553	0.032131	0.042828	0.017095	0.032053	0.113418	0.074317	0.000318	0.072384	0.109989	0.114482	0.014336	0.030803	0.043185	0.075124	0.138772	0.182119	0.756492
5	0.014601	0.091695	0.152795	0.014685	0.031495	0.066722	0.003593	0.009280	0.005766	0.091605	0.137774	0.070131	0.162982	0.067822	0.024784	0.022553	0.031718	0.165540	0.198352	0.829534

Figure 11. Optimal Portfolios with Asset Allocations

The first observation is an important point to consider when comparing portfolios. Return seems to hold a strong positive correlation with volatility. This makes it difficult to choose because of the risk-reward trade-off that is set in place. Having access to the Sharpe ratios suggests one way to decide between several different portfolios. The decision that an investor makes will ultimately depend on his or her risk tolerance and investment goals. Another observation of the data frame above is the varying composition of assets in each portfolio. Assets like BTC and ETH are known to be less volatile than assets like LTC and SAND. Assets with low volatility tend to hold more weight in low-risk portfolios than assets with high volatility. In high-risk portfolios, assets with high volatility appear to take more space. This is a natural consequence and a direct connection of the risk-reward trade-off mentioned above.

To gain more insight from our results, we decided to modify our results for a comparison with a benchmark in broader markets such as the S&P 500 index.

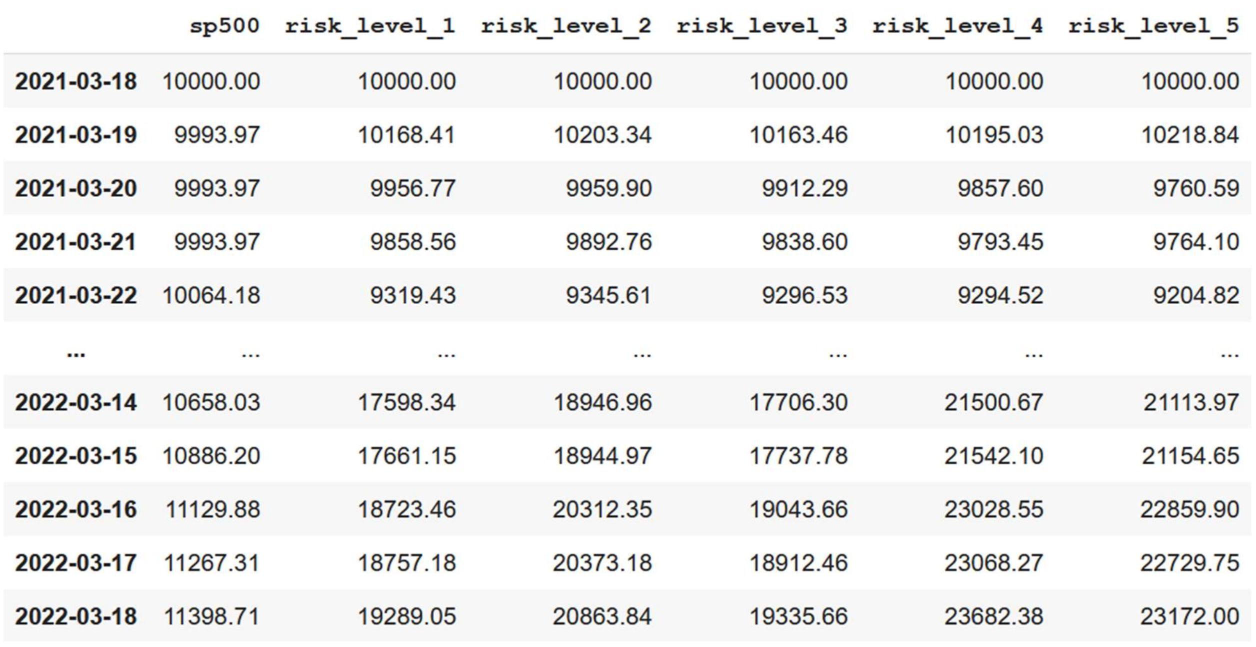
```
start = datetime.today()-relativedelta(years=1, days=2) # Analyze 1-year histori-
cal data
end = datetime.today()

sp500 = web.DataReader(['sp500'], 'fred', start, end)
sp500 = sp500.pct_change() # Daily returns

performances = [sp500] # Track performances

data_hist = data[data.index > start].pct_change() # Crypto assets historical prices

for ind, p in enumerate(portfolios):
  p_assets = p['assets']
  p_weights = [x['allocation'] for x in p_assets]

  dcopy = data_hist.copy()

  for i, col in enumerate(dcopy.columns):
    dcopy[col] = dcopy[col].apply(lambda x: x*p_weights[i])

  dcopy['risk_level_'+str(ind+1)] = dcopy.apply(lambda x: np.nansum(x)/sum([y if not m
ath.isnan(float(x[i])) else 0 for (i, y) in enumerate(p_weights)]), axis=1)

  performances.append(dcopy['risk_level_'+str(ind+1)])

index_ports_returns = pd.concat(performances, axis=1)
```

	sp500	risk_level_1	risk_level_2	risk_level_3	risk_level_4	risk_level_5
2021-03-18	NaN	NaN	NaN	NaN	NaN	NaN
2021-03-19	-0.000603	0.016841	0.020334	0.016346	0.019503	0.021884
2021-03-20	NaN	-0.020814	-0.023859	-0.024713	-0.033097	-0.044844
2021-03-21	NaN	-0.009864	-0.006742	-0.007434	-0.006508	0.000360
2021-03-22	0.007025	-0.054686	-0.055308	-0.055096	-0.050945	-0.057279
...	...	...	...	...	...	...
2022-03-14	-0.007421	0.038626	0.031277	0.033679	0.036147	0.033449
2022-03-15	0.021408	0.003569	-0.000105	0.001778	0.001927	0.001927
2022-03-16	0.022384	0.060150	0.072176	0.073621	0.069002	0.080609
2022-03-17	0.012348	0.001801	0.002995	-0.006889	0.001725	-0.005693
2022-03-18	0.011662	0.028356	0.024083	0.022377	0.026621	0.019457

Figure 12. Daily Returns of S&P 500 and Portfolios (03/18/2021 – 03/18/2022)

A starting balance of $10,000 is used to transform the daily returns into account balances. The historical performance tracks the price changes of the S&P 500 index alongside all five portfolios starting from March 18, 2021 and ending on March 18, 2022. Using daily returns, the closing balance at the end of each day is calculated in succession as seen below.

```
index_ports_returns.iloc[0] = 10000 # Starting balance
for enum, z in enumerate(index_ports_returns.columns):
  for i in range(1, len(index_ports_returns)):
    if np.isnan(index_ports_returns.iloc[i,enum]):
      index_ports_returns.iloc[i,enum] = index_ports_returns.iloc[i-
1,enum] # Take previous day balance if no data
    else:
      index_ports_returns.iloc[i,enum] = index_ports_returns.iloc[i-
1,enum] * (1+index_ports_returns.iloc[i,enum])

index_ports_returns = index_ports_returns.applymap(lambda x: round(x, 2))
```



	sp500	risk_level_1	risk_level_2	risk_level_3	risk_level_4	risk_level_5
2021-03-18	10000.00	10000.00	10000.00	10000.00	10000.00	10000.00
2021-03-19	9993.97	10168.41	10203.34	10163.46	10195.03	10218.84
2021-03-20	9993.97	9956.77	9959.90	9912.29	9857.60	9760.59
2021-03-21	9993.97	9858.56	9892.76	9838.60	9793.45	9764.10
2021-03-22	10064.18	9319.43	9345.61	9296.53	9294.52	9204.82
...	...	...	...	...	...	...
2022-03-14	10658.03	17598.34	18946.96	17706.30	21500.67	21113.97
2022-03-15	10886.20	17661.15	18944.97	17737.78	21542.10	21154.65
2022-03-16	11129.88	18723.46	20312.35	19043.66	23028.55	22859.90
2022-03-17	11267.31	18757.18	20373.18	18912.46	23068.27	22729.75
2022-03-18	11398.71	19289.05	20863.84	19335.66	23682.38	23172.00

Figure 13. Historical Performance With $10,000 Initial Balance

The data points from Figure 13 are plotted as shown. As expected, aggressive portfolios like Risk Level 4 and Risk Level 5 outperformed the other portfolios for most sessions over the past 12 months. In periods of high inflow and positive uptrend, the relative performance of these aggressive portfolios is much greater. However, the drawdown on these portfolios is typically worse during periods of decline.

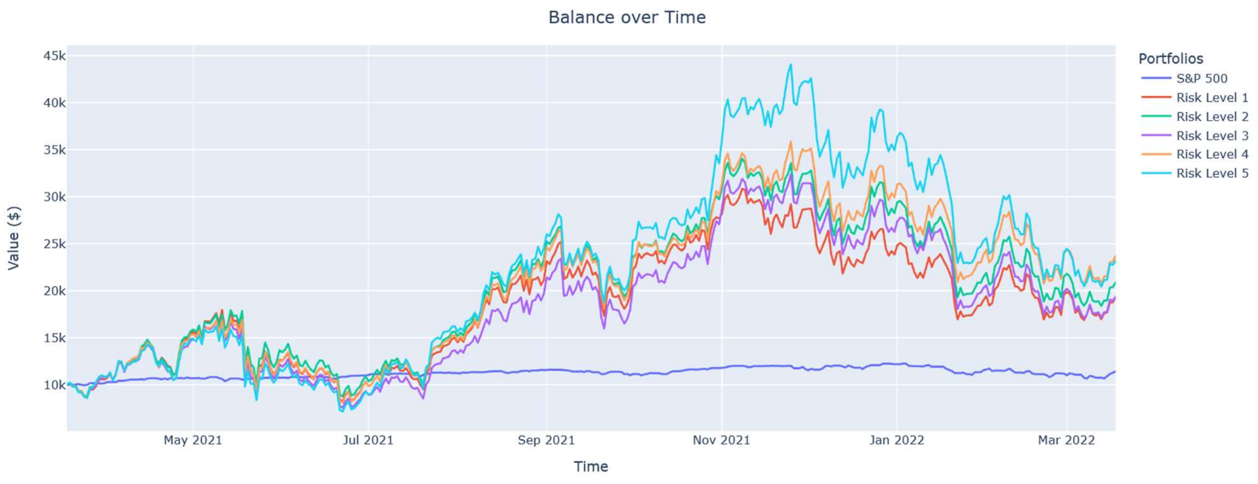
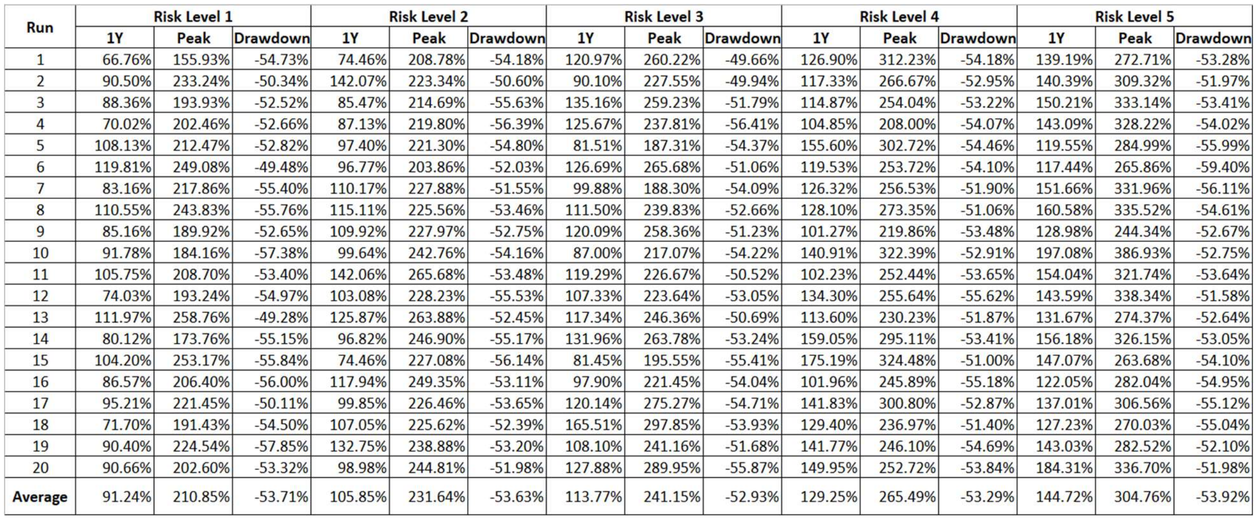


Figure 14. Historical Performance Comparison

Testing

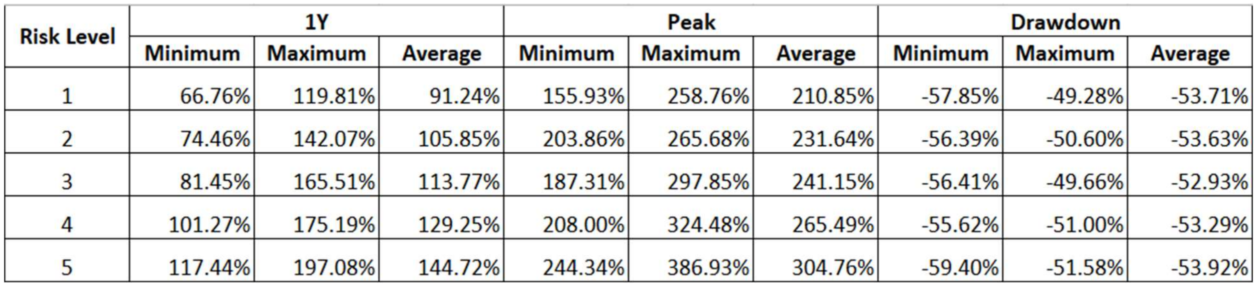
We ran a series of tests to confirm our assumptions of the relationship between volatility and return of portfolios. Figure 15 displays the complete set of results for each of the five risk levels over 20 iterations. For each iteration, we calculated the 1-year return, peak return, and drawdown for the corresponding portfolio in each risk bucket. The 1-year return is calculated by the percentage change between the starting and ending balances. The peak return is equal to the percentage change at the day of maximum portfolio balance. The drawdown column refers to the largest peak-to-trough decline in percentage over the observed past year.



Run	Risk Level 1			Risk Level 2			Risk Level 3			Risk Level 4			Risk Level 5		
	1Y	Peak	Drawdown	1Y	Peak	Drawdown	1Y	Peak	Drawdown	1Y	Peak	Drawdown	1Y	Peak	Drawdown
1	66.76%	155.93%	-54.73%	74.46%	208.78%	-54.18%	120.97%	260.22%	-49.66%	126.90%	312.23%	-54.18%	139.19%	272.71%	-53.28%
2	90.50%	233.24%	-50.34%	142.07%	223.34%	-50.60%	90.10%	227.55%	-49.94%	117.33%	266.67%	-52.95%	140.39%	309.32%	-51.97%
3	88.36%	193.93%	-52.52%	85.47%	214.69%	-55.63%	135.16%	259.23%	-51.79%	114.87%	254.04%	-53.22%	150.21%	333.14%	-53.41%
4	70.02%	202.46%	-52.66%	87.13%	219.80%	-56.39%	125.67%	237.81%	-56.41%	104.85%	208.00%	-54.07%	143.09%	328.22%	-54.02%
5	108.13%	212.47%	-52.82%	97.40%	221.30%	-54.80%	81.51%	187.31%	-54.37%	155.60%	302.72%	-54.46%	119.55%	284.99%	-55.99%
6	119.81%	249.08%	-49.48%	96.77%	203.86%	-52.03%	126.69%	265.68%	-51.06%	119.53%	253.72%	-54.10%	117.44%	265.86%	-59.40%
7	83.16%	217.86%	-55.40%	110.17%	227.88%	-51.55%	99.88%	188.30%	-54.09%	126.32%	256.53%	-51.90%	151.66%	331.96%	-56.11%
8	110.55%	243.83%	-55.76%	115.11%	225.56%	-53.46%	111.50%	239.83%	-52.66%	128.10%	273.35%	-51.06%	160.58%	335.52%	-54.61%
9	85.16%	189.92%	-52.65%	109.92%	227.97%	-52.75%	120.09%	258.36%	-51.23%	101.27%	219.86%	-53.48%	128.98%	244.34%	-52.67%
10	91.78%	184.16%	-57.38%	99.64%	242.76%	-54.16%	87.00%	217.07%	-54.22%	140.91%	322.39%	-52.91%	197.08%	386.93%	-52.75%
11	105.75%	208.70%	-53.40%	142.06%	265.68%	-53.48%	119.29%	226.67%	-50.52%	102.23%	252.44%	-53.65%	154.04%	321.74%	-53.64%
12	74.03%	193.24%	-54.97%	103.08%	228.23%	-55.53%	107.33%	223.64%	-53.05%	134.30%	255.64%	-55.62%	143.59%	338.34%	-51.58%
13	111.97%	258.76%	-49.28%	125.87%	263.88%	-52.45%	117.34%	246.36%	-50.69%	113.60%	230.23%	-51.87%	131.67%	274.37%	-52.64%
14	80.12%	173.76%	-55.15%	96.82%	246.90%	-55.17%	131.96%	263.78%	-53.24%	159.05%	295.11%	-53.41%	156.18%	326.15%	-53.05%
15	104.20%	253.17%	-55.84%	74.46%	227.08%	-56.14%	81.45%	195.55%	-55.41%	175.19%	324.48%	-51.00%	147.07%	263.68%	-54.10%
16	86.57%	206.40%	-56.00%	117.94%	249.35%	-53.11%	97.90%	221.45%	-54.04%	101.96%	245.89%	-55.18%	122.05%	282.04%	-54.95%
17	95.21%	221.45%	-50.11%	99.85%	226.46%	-53.65%	120.14%	275.27%	-54.71%	141.83%	300.80%	-52.87%	137.01%	306.56%	-55.12%
18	71.70%	191.43%	-54.50%	107.05%	225.62%	-52.39%	165.51%	297.85%	-53.93%	129.40%	236.97%	-51.40%	127.23%	270.03%	-55.04%
19	90.40%	224.54%	-57.85%	132.75%	238.88%	-53.20%	108.10%	241.16%	-51.68%	141.77%	246.10%	-54.69%	143.03%	282.52%	-52.10%
20	90.66%	202.60%	-53.32%	98.98%	244.81%	-51.98%	127.88%	289.95%	-55.87%	149.95%	252.72%	-53.84%	184.31%	336.70%	-51.98%
Average	91.24%	210.85%	-53.71%	105.85%	231.64%	-53.63%	113.77%	241.15%	-52.93%	129.25%	265.49%	-53.29%	144.72%	304.76%	-53.92%

Figure 15. Test run statistics

Figure 16 shows a compiled and more comprehensive view of the test runs. One observation that can be made is the strictly increasing values of 1-year returns as risk level is increased. This is not taken by surprise because portfolios that are known to carry more risk tend to offer potentially higher return. The peak returns also show similar trends of increasing return. These values suggest the possible range of price movements on the upside over the past year. The drawdown, on the other hand, shows the possible change in the value of an investment. For each risk level, the decline appears to be in a tighter range compared to return. This outcome is generally ideal for an investor because it means that the downside of a risky portfolio is not as significant over the long run. The average drawdown of Risk Level 3 is -52.93%, meaning that the average peak-to-trough decline is less than the drawdowns of Risk Levels 1 and 2. This may entirely be a coincidence given the small difference between the values. However, it is important to keep note of this observation as it raises the question of whether a low-risk portfolio is always the safest option.



Risk Level	1Y			Peak			Drawdown		
	Minimum	Maximum	Average	Minimum	Maximum	Average	Minimum	Maximum	Average
1	66.76%	119.81%	91.24%	155.93%	258.76%	210.85%	-57.85%	-49.28%	-53.71%
2	74.46%	142.07%	105.85%	203.86%	265.68%	231.64%	-56.39%	-50.60%	-53.63%
3	81.45%	165.51%	113.77%	187.31%	297.85%	241.15%	-56.41%	-49.66%	-52.93%
4	101.27%	175.19%	129.25%	208.00%	324.48%	265.49%	-55.62%	-51.00%	-53.29%
5	117.44%	197.08%	144.72%	244.34%	386.93%	304.76%	-59.40%	-51.58%	-53.92%

Figure 16. Compiled statistics

Conclusion

The MPT algorithm is an effective approach of analyzing the risk and return associated with a given portfolio. It relies on calculations using historical data and is flexible with the study timeframe and risk measure types. For the problem that we are attempting to solve, which mainly concerns the development of a balanced portfolio of crypto assets, the current model is suitable. This is because the directional movement of assets in the crypto market has a strong positive correlation. In other words, it is not common for two portfolios to behave differently in terms of direction. As shown in Figure 14, all five portfolios display similar trends, only differentiating in magnitude.

There are some disadvantages to the current model. First, there is no regard for forecasting in the algorithm design. One of the ways an investor makes investment decisions is to look at a particular asset and choosing to buy, hold, or sell it based on its current state and an approximate prediction on its future price. By only using historical returns as the main source of data, it is not possible to make any statement on what will happen to a basket of assets within the next few months or years. Another problem with the current model is its inflexibility with varying investment time horizons. Because our algorithm currently uses monthly returns, it is most appropriate to use as a decision support tool for medium to long-term investors.

This research used mean-variance analysis on the historical data of crypto assets to produce multiple portfolios for various risk tolerance levels. The current baseline model provides well-diversified investment ideas for all types of investors. The results generated are strongly supported by a sturdy mathematical model of multivariate analysis. There are ideas such as the points described above that will help to solidify the algorithm even further and allow expansion of our platform to offer a wider variety of features to investors.

References

https://www.investopedia.com/terms/m/modernportfoliotheory.asp

https://www.quantconnect.com/tutorials/introduction-to-financial-python/modern-portfolio-theory

https://www.analyticsvidhya.com/blog/2021/04/portfolio-optimization-using-mpt-in-python/

https://cryptobriefing.com/modern-portfolio-theory-and-the-efficient-crypto-portfolio/

https://medium.com/geekculture/optimal-cryptocurrencies-portfolio-allocation-with-modern-portfolio-theory-in-python-66a0dc98ed65

https://www.investopedia.com/articles/07/sharpe_ratio.asp

https://corporatefinanceinstitute.com/resources/knowledge/finance/portfolio-variance/